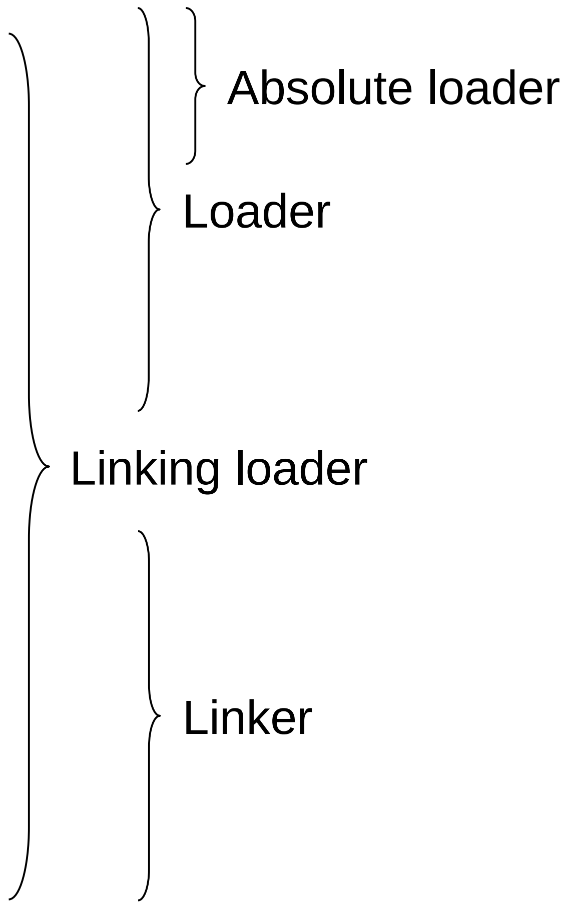
Chapter 3 Loaders and Linkers

Processes to Run an Object Program

- Loading
  - Brings object program into memory
- Relocation
  - Modifies the object program where absolute addresses are specified
- Linking
  - Combines two or more separate object programs and supplies information needed to allow cross-references.

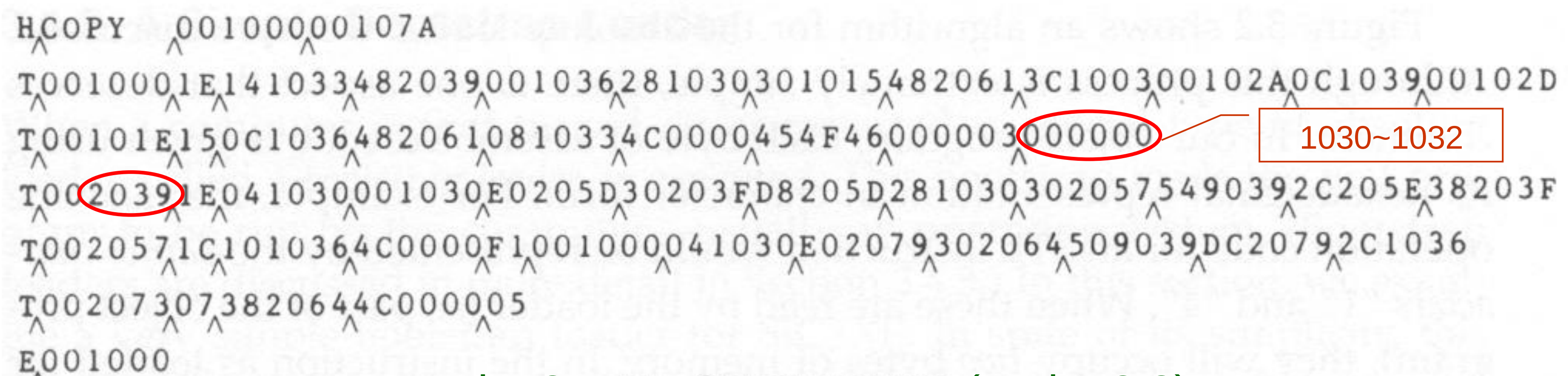


Absolute Loader

for SIC Machine

Absolute Loader

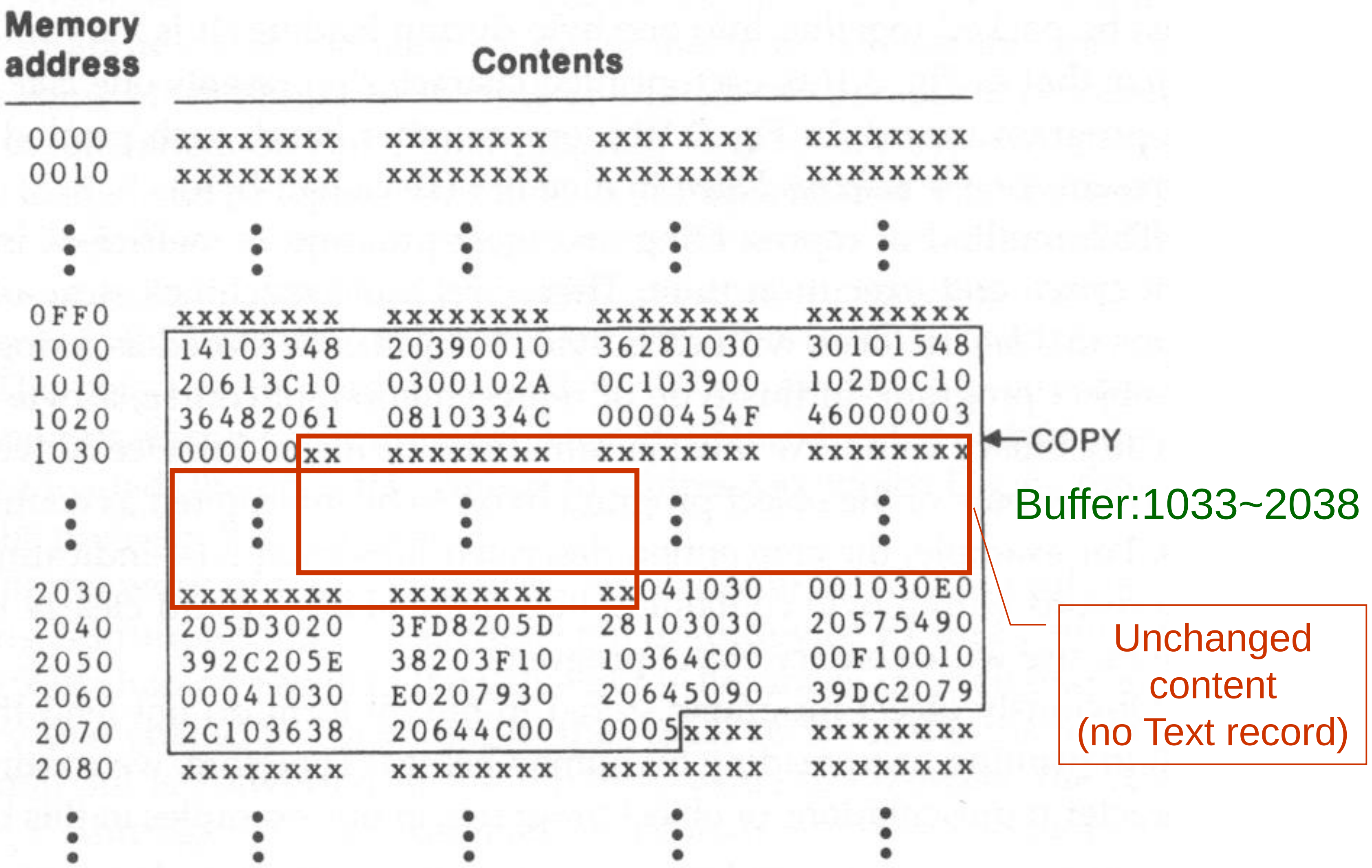
- In a single pass
  - Check the Header record for program name, starting address, and length
  - Bring the object program contained in the Text record to the indicated address
  - No need to perform program linking and relocation
  - Start the execution by jumping to the address specified in the End record



```
H^COPY  ^001000^00107A
T^001000^1E^141033^482039^001036^281030^301015^482061^3C1003^00102A^0C1039^00102D
T^00101E^15^0C1036^482061^081033^4C0000^454F46^000003^000000
T^002039^1E^041030^001030^E0205D^30203F^D8205D^281030^302057^549039^2C205E^38203F
T^002057^1C^101036^4C0000^F1^001000^041030^E02079^302064^509039^DC2079^2C1036
T^002073^07^382064^4C0000^05
E^001000
```

Fig. 3.1 **(a) Object program** (= Fig. 2.3)

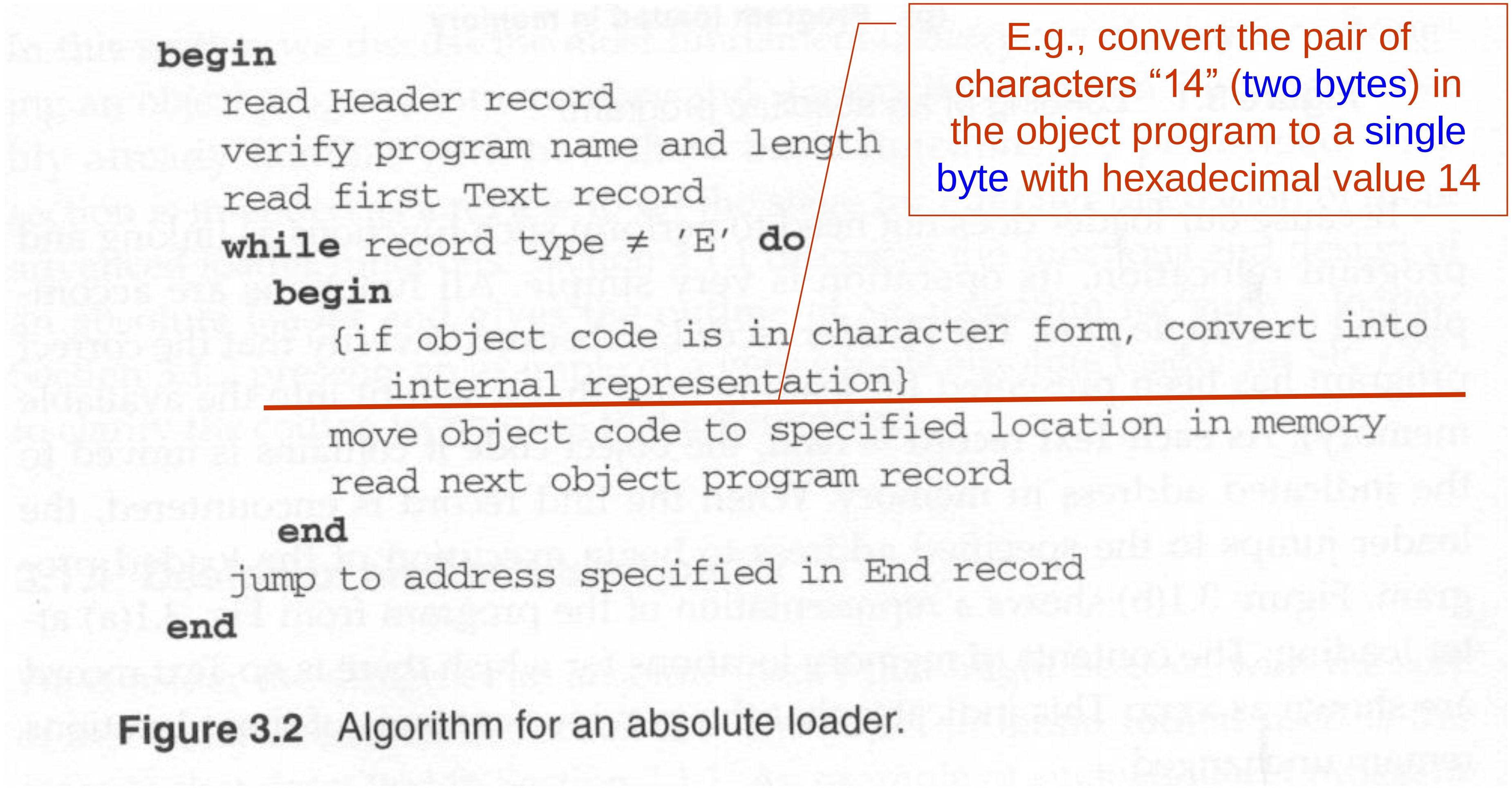
Program Loaded in Memory



Memory address	Contents			
0000	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
0010	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
⋮	⋮	⋮	⋮	⋮
0FF0	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
1000	14103348	20390010	36281030	30101548
1010	20613C10	0300102A	0C103900	102D0C10
1020	36482061	0810334C	0000454F	46000003
1030	000000xx	xxxxxxxx	xxxxxxxx	xxxxxxxx
⋮	⋮	⋮	⋮	⋮
2030	xxxxxxxx	xxxxxxxx	xx041030	001030E0
2040	205D3020	3FD8205D	28103030	20575490
2050	392C205E	38203F10	10364C00	00F10010
2060	00041030	E0207930	20645090	39DC2079
2070	2C103638	20644C00	0005xxxx	xxxxxxxx
2080	xxxxxxxx	xxxxxxxx	xxxxxxxx	xxxxxxxx
⋮	⋮	⋮	⋮	⋮

Fig. 3.1 **(b) Program loaded in memory**

Algorithm for an Absolute Loader (Fig. 3.2)



```
begin
    read Header record
    verify program name and length
    read first Text record
    while record type ≠ 'E' do
      begin
        {if object code is in character form, convert into
           internal representation}
        move object code to specified location in memory
        read next object program record
      end
    jump to address specified in End record
end
```

Figure 3.2 Algorithm for an absolute loader.

E.g., convert the pair of characters “14” (two bytes) in the object program to a single byte with hexadecimal value 14

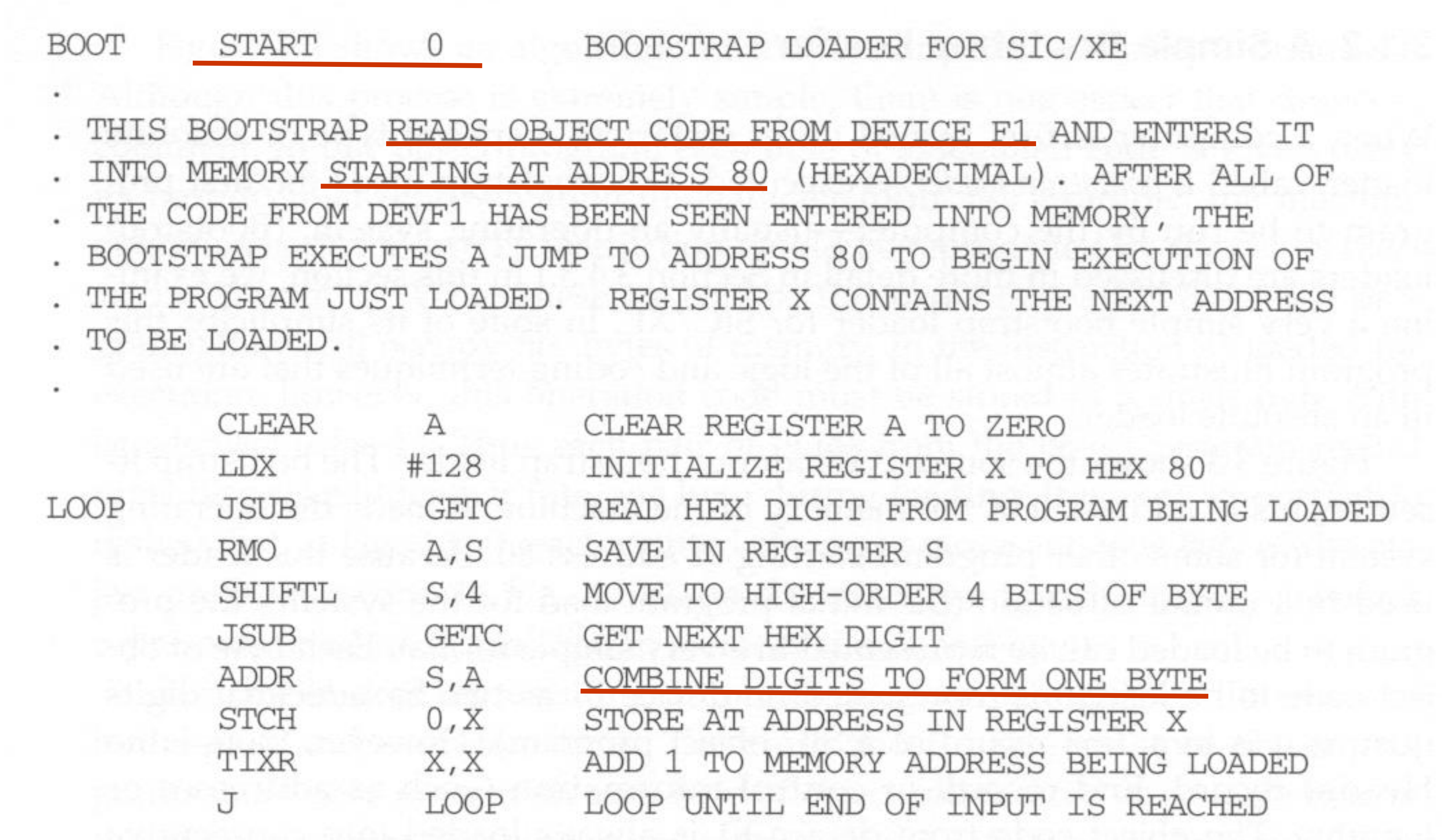
A Simple Bootstrap Loader

for SIC/XE Machine

Bootstrap Loader

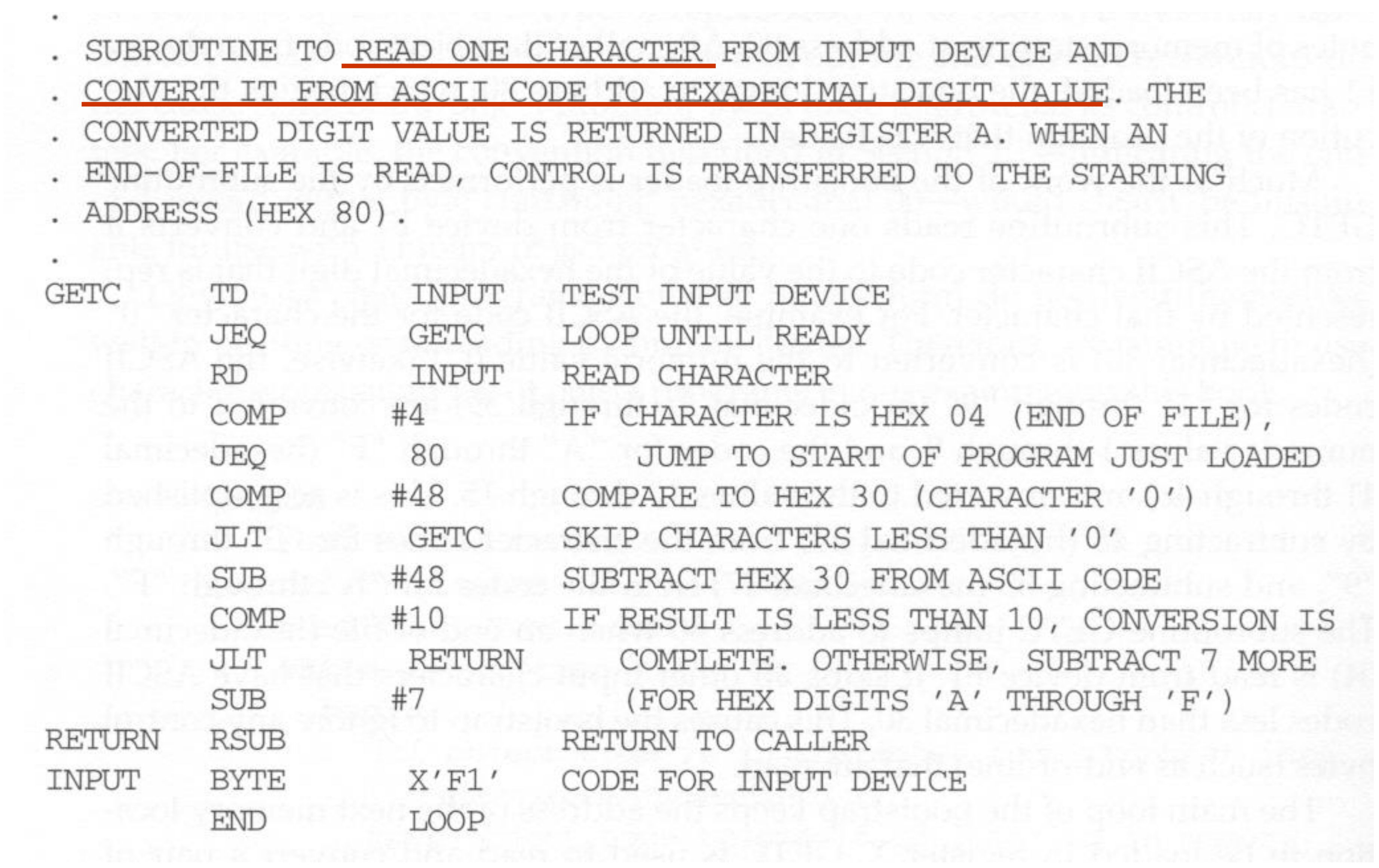
- A special type of absolute loader
- Executed when a computer is turned on or restarted
- Begins at address 0 in the memory
- Loads the first program (usually an OS) from a specific device (e.g., device F1)
- Load a very simple format of object program (no Header and End records or control information)
- Load the program into consecutive bytes of memory, starting at a specific address
- Jumps to the starting address to execute the program after all of the objected code has been loaded

A Simple Bootstrap Loader for SIC/XE (Fig. 3.3)



```
BOOT    START   0         BOOTSTRAP LOADER FOR SIC/XE
.
. THIS BOOTSTRAP READS OBJECT CODE FROM DEVICE F1 AND ENTERS IT
. INTO MEMORY STARTING AT ADDRESS 80 (HEXADECIMAL). AFTER ALL OF
. THE CODE FROM DEVF1 HAS BEEN SEEN ENTERED INTO MEMORY, THE
. BOOTSTRAP EXECUTES A JUMP TO ADDRESS 80 TO BEGIN EXECUTION OF
. THE PROGRAM JUST LOADED.  REGISTER X CONTAINS THE NEXT ADDRESS
. TO BE LOADED.
.
        CLEAR   A         CLEAR REGISTER A TO ZERO
        LDX    #128       INITIALIZE REGISTER X TO HEX 80
LOOP    JSUB    GETC      READ HEX DIGIT FROM PROGRAM BEING LOADED
        RMO     A,S       SAVE IN REGISTER S
        SHIFTL  S,4       MOVE TO HIGH-ORDER 4 BITS OF BYTE
        JSUB    GETC      GET NEXT HEX DIGIT
        ADDR    S,A       COMBINE DIGITS TO FORM ONE BYTE
        STCH    0,X       STORE AT ADDRESS IN REGISTER X
        TIXR    X,X       ADD 1 TO MEMORY ADDRESS BEING LOADED
        J       LOOP      LOOP UNTIL END OF INPUT IS REACHED
```

A Simple Bootstrap Loader for SIC/XE



```
.
. SUBROUTINE TO READ ONE CHARACTER FROM INPUT DEVICE AND
. CONVERT IT FROM ASCII CODE TO HEXADECIMAL DIGIT VALUE. THE
. CONVERTED DIGIT VALUE IS RETURNED IN REGISTER A. WHEN AN
. END-OF-FILE IS READ, CONTROL IS TRANSFERRED TO THE STARTING
. ADDRESS (HEX 80).
.
GETC     TD       INPUT    TEST INPUT DEVICE
         JEQ      GETC     LOOP UNTIL READY
         RD       INPUT    READ CHARACTER
         COMP     #4       IF CHARACTER IS HEX 04 (END OF FILE),
         JEQ      80          JUMP TO START OF PROGRAM JUST LOADED
         COMP     #48      COMPARE TO HEX 30 (CHARACTER '0')
         JLT      GETC     SKIP CHARACTERS LESS THAN '0'
         SUB      #48      SUBTRACT HEX 30 FROM ASCII CODE
         COMP     #10      IF RESULT IS LESS THAN 10, CONVERSION IS
         JLT      RETURN      COMPLETE. OTHERWISE, SUBTRACT 7 MORE
         SUB      #7          (FOR HEX DIGITS 'A' THROUGH 'F')
RETURN   RSUB              RETURN TO CALLER
INPUT    BYTE     X'F1'    CODE FOR INPUT DEVICE
         END      LOOP
```